

Автоматический адаптивный решатель СЛАУ для гибридных систем

Химич А.Н.¹, Чистякова Т.В.¹, Баранов А.Ю.¹

¹Институт кибернетики им. В.М. Глушкова, просп. Глушкова, 40, Киев, Украина

dept150@insyg.kiev.ua

Аннотация. В статье представляется методология создания программного средства для исследования и решения систем линейных алгебраических уравнений на гибридной системе, сочетающей вычисления на многоядерных компьютерах MIMD-архитектуры с ускорением вычислений на графических процессорах SIMD-архитектуры. Предлагается автоматический решатель по исследованию и решению систем линейных алгебраических уравнений с приближенно заданными исходными данными на гибридных системах с функцией адаптивной настройки алгоритма, программы и архитектуры компьютера на свойства решаемой задачи. При этом пользователи освобождаются от самостоятельного выбора алгоритма решения в соответствии со свойствами решаемой задачи, анализа достоверности получаемых результатов, а также проблем, связанных с программированием на сложных гибридных системах

Ключевые слова

Гибридная система, многоядерные компьютеры MIMD-архитектуры, графический процессор, автоматический адаптивный решатель, система линейных алгебраических уравнений, приближенно заданные исходные данные

1 Введение

В настоящее время наблюдается определенная тенденция в развитии вычислительных систем. С одной стороны, как и прежде, продолжается рост производительности компьютеров за счет увеличения числа процессорных ядер. С другой стороны, становятся все более популярными гибридные системы, гетерогенные системы, высокая производительность которых обусловлена применением вычислительных элементов принципиально новой архитектуры. Первое направление развития требует от алгоритмов большой степени параллелизма на однотипных процессорных ядрах, который на программном уровне решается с помощью специальных программных систем таких, например, как MPI. Второе направление требует от алгоритма более сложной многоуровневой параллельной модели. Программа для гибридной системы, CPU с графическим ускорителем (GPU), состоит из кода для CPU, который может быть написан на обычном языке программирования, например, на Си, C++, и кода для графического процессора, написанного на специальном языке, например CUDA. Если же рассматривается гибридная система с многоядерными CPU и многими GPU, то очень остро стоит вопрос не только в обеспечении высокой производительности отдельных CPU и GPU, но также об эффективном использовании вычислительных компонентов при решении конкретной задачи. Возникает необходимость в планировании вычислений на вычислительных ресурсах гибридной системы с целью как можно быстрее получить решение задачи.

Известно, что многие программные приложения из различных предметных областей науки и производства зависят от успешного решения систем линейных алгебраических уравнений (СЛАУ), к которым сводятся решаемые задачи. Несмотря на большое внимание к созданию программного обеспечения по линейной алгебре на различные типы компьютерных архитектур [1, 2], многие проблемы эффективного его использования остаются.

Прежде всего, при использовании современного программного обеспечения для решения СЛАУ возникают проблемы решения задач с приближенно заданными исходными данными:

- исследование корректности постановки математической задачи в компьютере;
- определение обусловленности компьютерной модели задачи;
- оценка погрешности получаемого компьютерного решения.

Другая проблема связана с эффективным использованием вычислительных ресурсов компьютеров гибридной архитектуры, с согласованием распределения вычислительных ресурсов на ядрах CPU и GPU, а также оптимизацией коммуникационных расходов между ядрами CPU и GPU. Кроме того, пользователю нужно осваивать новые языки программирования для написания программ на различные вычислительные ресурсы.

Предлагается автоматический адаптивный решатель по исследованию и решению СЛАУ на гибридных системах, реализующий автоматическое исследование и решение задачи с приближенно заданными исходными данными, анализ получаемых решений, осуществляя при этом динамическое планирование вычислений на каждом шаге алгоритма на необходимых вычислительных ресурсах.

2 Реализация автоматического исследования и решения задачи с автоматическим планированием вычислений на гибридной системе

Основные концептуальные принципы автоматического адаптивного решателя для СЛАУ на гибридных системах:

- возможность решения задач с приближенно заданными исходными данными;
- естественные для пользователя формы ввода исходных данных;
- автоматизация процессов компьютерного исследования математических свойств компьютерной модели задачи, выбора алгоритма и синтеза программы решения;
- решение задачи с оценкой достоверности получаемых компьютерных результатов;
- получение не только решения задачи, но и протокола процесса решения задачи с анализом выявленных ее свойств и достоверности полученных результатов;
- реализация «скрытого» планирования вычислений на требуемые вычислительные ресурсы для эффективного решения задачи.

«Скрытое» планирование вычислений предполагает двухуровневое планирование вычислений:

1-й уровень - автоматическое исследование математических свойств компьютерной модели задачи и выбор соответствующего алгоритма решения с оценками достоверности (оценка наследственной погрешности в математическом решении и оценка вычислительной погрешности) [3];

2-й уровень - автоматическое планирование реализации конкретного алгоритма на предоставленные вычислительные ресурсы гибридной системы с целью наименьших затрат компьютерного времени и мощностей гибридных компонентов.

В данном программном обеспечении реализованы известные алгоритмы прямых методов Холецкого и Гаусса - для решения СЛАУ с плотными невырожденными матрицами, метод наименьших квадратов - для нахождения нормального обобщенного решения систем с матрицами неполного ранга. Выбор необходимого алгоритма решения задачи и синтез необходимой программы происходит в процессе исследования ее характерных свойств в компьютере (вид матрицы системы, положительная определенность матрицы, вырожденность матрицы, плохая обусловленность и т. д.) с учетом приближенного характера исходных данных.

Для реализации планирования вычислений второго уровня используется система STARPU [4], которая позволяет планировать выполнение задачи с использованием многоядерных CPU и GPU в различном сочетании, наиболее эффективном для конкретной задачи. Параллелизм задачи осуществляется используя последовательные коды программ, с указанием частей алгоритмов (подзадач), которые реализуются на различных вычислительных компонентах гибрида: только на ядре CPU, только на GPU и на обоих ресурсах одновременно (гибридное задание). STARPU динамически планирует вычисления с учетом затрат

времени выполнения каждой подзадачи на каждом доступном вычислительном устройстве. При этом автоматически выполняется асинхронное копирование необходимых данных как между CPU и GPU, так и между GPU. Исходные данные задачи один раз регистрируются в среде StarPU, и в дальнейшем система сама будет копировать необходимые данные между CPU и GPU, максимально перекрывая время выполнения коммутационных связей временем выполнения математических операций.

На рис. 1 представлен фрагмент схемы исследования и решения СЛАУ данным решателем.

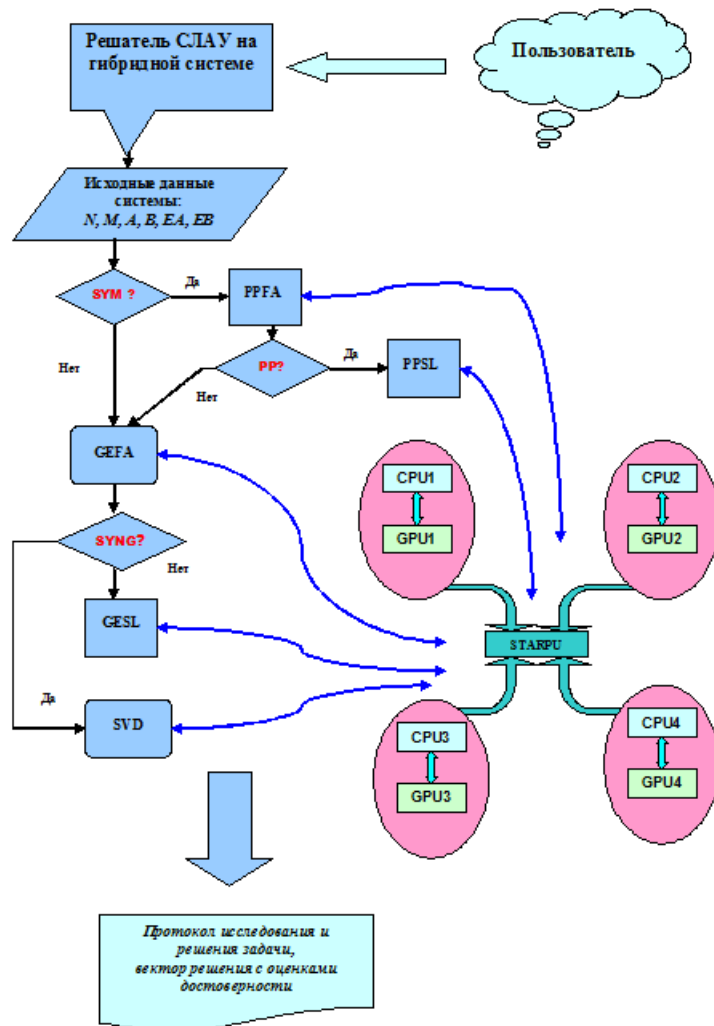


Рис. 1. Схема автоматического исследования и решения СЛАУ

Здесь можно выделить следующие этапы работ по исследованию и решению задачи:

- 1) Пользователю предоставляются различные формы ввода исходных данных СЛАУ: количество строк (N) и столбцов (M) матрицы, элементы матрицы (A) и максимальная относительная погрешность их задания (EA), элементы матрицы правых частей (B) и максимальная относительная погрешность их задания (EB), а также указывается вид матрицы – симметричная или нет.
- 2) Если матрица системы симметричная, то для исследования ее на положительную определенность выбирается метод Холецкого (программа PPFA). При этом с помощью STARPU определяются необходимые вычислительные ресурсы и выбирается соответствующая программа для реализации алгоритма решения: только на ядрах CPU, на одном ядре CPU с использованием ускорителя GPU, на нескольких ядрах CPU с использованием ускорителей GPU.
- 3) Если матрица системы в компьютере оказалась положительно определенной, то решение СЛАУ продолжается (программа PPSL с подключением STARPU для автоматизации планирования вычислений 2-го уровня), иначе матрица исследуется на вырожденность методом Гаусса (программа GEFA с подключением STARPU).

- 4) Если матрица системы в компьютере оказалась невырожденной, то решение СЛАУ продолжается (программа GESL с подключением STARPU), иначе вычисляется нормальное обобщенное решение методом наименьших квадратов (программа SVD с подключением STARPU).
- 5) По окончанию вычислительного процесса выдается протокол решения задачи с указанием методов, которыми исследовалась задача, задействованных вычислительных ресурсов, выявленных свойств, оценок достоверности. Решение СЛАУ пользователь может сохранить по желанию на различных носителях информации.

Рассмотрим, как реализуется планирование вычислений второго уровня с помощью StarPU на примере алгоритма факторизации $n \times n$ симметричной положительно определенной матрицы системы методом Холецкого: $A = LL^T$.

За основу вычислительной схемы берем схему реализации блочного алгоритма Холецкого на основе последовательных кодов четырех основных операций из пакетов программ LAPACK [5] и BLAS [6]:

xPOTRF - факторизация ведущего диагонального блока матрицы;

xSYRK - модификация диагональных (нижних треугольных) блоков матрицы;

xGEMM - преобразование поддиагональных блоков матрицы, которые находятся справа от ведущего блока;

xTRSM - преобразование блоков матрицы, которые находятся в одном столбике с ведущим блоком (решение СЛАУ с треугольной матрицей).

Здесь вместо x указывается необходимая разрядность вычислений: S - одинарная разрядность, D - удвоенная разрядность.

Псевдокод реализации $A = LL^T$ -факторизации матрицы методом Холецкого имеет вид:

for($k = 0; k < Nt; k++$)

$A[k][k] \leftarrow DPOTRF(A[k][k])$

for($m = k + 1; m < Nt; m++$)

$A[m][k] \leftarrow DTRSM(A[k][k], A[m][k])$

for($n = k + 1; n < Nt; n++$)

$A[n][n] \leftarrow DSYRK(A[n][k], A[n][n])$

for($m = n + 1; m < Nt; m++$)

$A[m][n] \leftarrow DGEMM(A[m][k], A[n][k], A[m][n])$

Как мы видим из псевдокода, операция во второй строке может быть выполнена раньше, чем был полностью выполнен предыдущий шаг в цикле. Для начала ее выполнения достаточно, чтобы на предыдущем шаге было определено $A[k][k]$. Аналогичные рассуждения можно привести и для остальных операций. Таким образом, чтобы приступить к выполнению операций на шаге k цикла, не обязательно, чтобы шаг $k-1$ был завершен. Достаточно, чтобы были получены необходимые для каждой конкретной операции блоки матрицы. StarPU дает возможность заранее объявить все операции, которые будут нужны для выполнения алгоритма, а также определить между ними зависимости, которые дают возможность контролировать порядок выполнения операций. И затем все операции будут выполняться в том порядке и на том количестве процессоров, которые обеспечат самую высокую эффективность реализации алгоритма.

В итоге, алгоритм состоит из выполнения следующих процедур:

- разделение задачи на подзадачи;
- параллельные подзадачи (xTRSM, xDSYRK, xDGEMM) "планируются" для эффективного выполнения на GPU с учетом асинхронности, величины блока и т. д.;
- последовательная подзадача xPOTRF – разложение диагонального блока выполняется на CPU;
- малые задания на CPU частично покрываются большими заданиями GPU (умножение матриц).

Таким образом, алгоритм Холецкого можно представить в виде направленного ациклического графа, который представлен на рис. 2.

В этом графе вершины представляют собой операции, которые нужно выполнить, а направленные ребра - зависимости между ними. Для каждой последующей подзадачи необходимые данные будут готовы тогда и только тогда, когда будут выполнены все предыдущие подзадачи, от которых она зависит. Таким образом, для решения СЛАУ методом Холецкого нужно выполнить эти операции в определенной последовательности.

StarPU предоставляет набор планировщиков, с помощью которых можно добиться максимально быстрой реализации алгоритма в зависимости от архитектуры гибридной системы. Для алгоритма Холецкого самым эффективным является Heft-планировщик, основанный на Heterogeneous Earliest Finish Time (HEFT) [7]. Для каждого доступного вычислительного устройства этот планировщик подсчитывает величину $Avail(P_i)$, определяющую в какое время каждое из вычислительных устройств может быть доступным, т. е. все подзадачи, которые на нем выполнялись, завершатся. Причем, новая подзадача T будет выполняться на том вычислительном устройстве, которое минимизирует время ее завершения.

В зависимости от ожидаемого времени выполнения $Est(T)$, это условие можно выразить следующей формулой:

$$\min_{P_i} (Avail(P_i) + Est(T)) \quad (1)$$

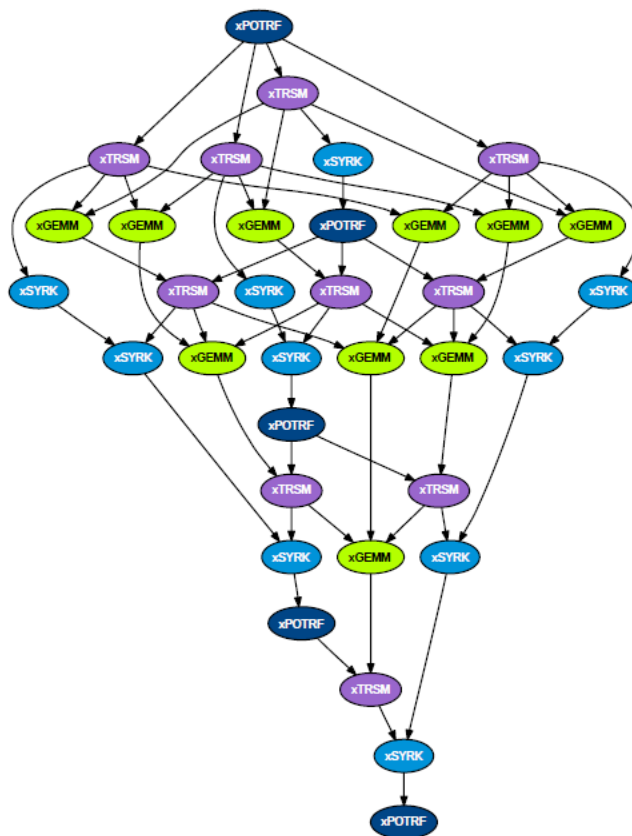


Рис. 2. Графическое представление алгоритма метода Холецкого

Разработчику алгоритма требуется задать функции, которые возвращают ожидаемое время выполнения задачи $Est(T)$. Для рассматриваемых операций оценка времени выполнения может быть легко получена, так как на каждом шаге алгоритма время выполнения каждой операции зависит только от порядка матриц. StarPU предоставляет структуру `starpu_perfmodel_t`, полями которой являются указатели на функции, вычисляющие время выполнения подзадач. Таким образом, имеется возможность заранее задавать модель, которая описывает время выполнения задач. Также StarPU позволяет задавать модель, которая базируется на истории запуска задач на предыдущих шагах. Например, в случае с ал-

горитмом Холецкого, время выполнения факторизации диагонального блока можно брать таким, каким оно было получено на самом первом шаге.

Таким образом, автоматическое исследование свойств задачи и построение соответствующей вычислительной схемы, а также синтез программы решения при автоматическом планировании выполнения подзадач, на которые разделяется задача, на требуемые ресурсы гибридной системы позволяют при минимальных затратах вычислительных ресурсов и компьютерного времени получить решение с оценками его достоверности.

3 Заключение

Многие научно-технические задачи сводятся к решению систем линейных алгебраических уравнений большой размерности. Для эффективного решения таких больших задач требуется эффективное использование ресурсов одновременно и CPU, и GPU. В условиях приближенных исходных данных свойства компьютерных моделей априори не известны. Поэтому необходимы новые подходы в создании программного обеспечения на гибридные системы, которые бы обеспечивали пользователей достоверным компьютерным решением при эффективном использовании вычислительных ресурсов сложных гибридных систем.

Эту цель реализует автоматический решатель СЛАУ с функцией адаптивной настройки алгоритма, программы и архитектуры гибридной системы на свойства решаемой задачи для ее эффективного решения с оценкой достоверности компьютерных результатов.

Список литературы

- [1] ScaLAPACK Home Page. The ScaLAPACK (or Scalable LAPACK) library includes a subset of LAPACK routines redesigned for distributed memory MIMD parallel computers. http://www.netlib.org/scalapack/scalapack_home.
- [2] CULA, GPU-accelerated linear algebra library. <http://www.culatools.com/>.
- [3] Химич А.Н., Молчанов И.Н., Попов А.В., и др. Параллельные алгоритмы решения задач вычислительной математики. -*Киев: Наук. думка, 2008.* – 248 с.
- [4] C. Augonnet, S. Thibault, R. Namyst, P. Wacrenier. StarPU: A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures. *Concurrency and Computation: Practice and Experience, Euro-Par 2009 best papers issue, 2010. Accepted for publication.*
- [5] LAPACK - Linear Algebra PACKage <http://www.netlib.org/lapack/>
- [6] BLAS (Basic Linear Algebra Subprograms. <http://www.netlib.org/blas/>
- [7] H. Topcuoglu, S. Hariri, and Min-You Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *Parallel and Distributed Systems, IEEE Transactions on, 13(3):260-274, 2002.*